set 340 contains both a send queue 222 and a receive queue 220, both types of hash entries resolve into a single queue control structure 330 that contains a pointer to the QDIO defined queues

[0029] The source queue hash table registration is as follows:

a. Obtain the exclusive update lock 402 for the hash tables. Updates to both types of hash tables can be serialized with a single lock.
b. Using the LPAR-ID.SUBCHANNEL# as key into the source hash table 310, determine the appropriate duplicate list header location 311 in the source queue hash table 310.
c. Once found, use the pointers 413 and 414 in a well known fashion to scan all hash key duplicate entries for an exact match with the LPAR-ID.SUBCHANNEL# being added. If found, then return the Duplicate Found error return to the TCP stack for the error to be dealt with there.
d. If there are no duplicates, at 312, add an entry to the source queue hash table 310.
e. Create the queue control 330 that is to be associated with the newly created entry.
f. Release the exclusive update lock 402 for the hash tables.

[0030] The target queue hash table registration is as follows:

a. Obtain exclusive lock 402 for the hash tables. Again, updates to both types of hash tables can be serialized with a single lock.
b. Using the target IP address as the key, determine the appropriate duplicate list header location in the target queue hash table 321.
c. Once found, use the pointers 423 and 424 in a well known fashion to scan all hash key duplicates for an exact match with the target IP addresses being added. If a duplicate is found, then return a Duplicate Found error to the TCP stack for the error to be handled there.
d. If no duplicates are found, at 322, add an entry to the target queue hash table 321.

e. Using the LPAR-ID.SUBCHANNEL# from the input, perform a search of the source queue hash table 310 to find the previously defined queue control 330 that is to be associated with the newly created entry. The control 330 contains the addresses to the I/O completion vector polling bytes (620a, 615a, and 612) that are used to pass initiative to the target.

f. Release the exclusive update lock 402 for the hash tables.

[0031] A send operation to send data from one LPAR partition to another is as follows:

a. As part of the processing of a socket API, the device driver 218 (software) modifies the send queue 440 (shown as downward arrow 222 in Fig. 2) to prime it with data to be transferred.

b. A send is initiated by a SIGA instruction to the device driver 218. This SIGA instruction explained in the aforementioned 09/253,246 application includes the subchannel number associated with the send queue 222.

c. The IQDIO indicator 228 of the subchannel control block 227 for the designated subchannel indicates that this is a IQDIO subchannel and that the send operation is to use the queue set 340 associated with this subchannel.

d. The shared serialization lock 401 is obtained for the queue lookup table 224 access.

e. The LPAR-ID from which the SIGA instruction is issued and the subchannel number in the instruction is used to build the LPAR-ID.SUBCHANNEL# key into the source hash table 310.

f. Obtain the outbound lock 431 to obtain exclusive serialization of the queue control 130 for the located entry in the source hash table 310.

g. Search the SLSB 442 to find the primed outbound storage buffer access list (SBAL) (shown as the buffer pointer 441) which points to the storage buffer access list element (SBALE) describing the packet of data to be moved to the target IP address.

h. Using the located SBAL, extract the destination IP address 244 from the outbound user buffer 443.

i. Use the IP address 244 to search the target queue hash table 320 to find the table entry 322 for the queue descriptor of the receive queue 220/445.

j. Obtain the inbound lock 432 to obtain exclusive serialization of the queue control 330 associated with the located target hash table entry 322.

k. The SLSB 447 of the receive queue 445 is searched to find an empty SBAL to receive the data.

l. Move the data in user buffer 443 of the send queue 440 to the user buffer 448 of the receiver queue 445 using internal millicode mechanism that overrides the normal restrictions on data moves between storage addresses in different LPAR partitions.

m. Update the SLSB 442 of the send queue 440 and the SLSB 447 of the receive queue 445. These updates are visible to the software and allows program manipulation of the send and receive queues 222 and 220.

n. Release the shared serialization lock 401.

o. Set a program initiative (I/O completion vector polling bytes - 620a, 615a, and 612) for the partition that contains the receive queue 220 to indicate that new elements or data are available on the receive queue 220. Having been thus informed, software in the target partition may process the data in its receive queue 220. Fig. 10 illustrates one inbodiment of the present invention wherein such an initiative for an I/O event is established.

p. Algorithmically determine if I/O interrupt generation is required, and if so generate the interrupt.

[0032] It will be understood that in the present embodiment, steps b-p of the send operation are performed by hardware, making the performance of these steps very reliable and at hardware speed. However, these steps, or some portion of them, could be done in software, if desired. This invention may also be used to transfer data between multiple virtual servers within a single partition.

[0033] Fig. 10 illustrates a three tiered hierarchy 600 of I/O completion vectors 610, 611 and 612. At the very top of the hierarchy, is a single global summary byte 612. Byte 612 is polled by